**Benemérita Universidad Autónoma de Puebla**



**EJERCICIO 8:**

**“Exponente”**

**Fac. de Cs. Computación**

**Asignatura:**

**Programación II**

**Profesora: Carmen Cerón Garnica**

**Equipo 4:**

**Emmanuel Flores Navarro 201929142**

**Erick Brian Ramírez Morales 201955520**

**07 / Mayo / 2021**

**INTRODUCCIÓN**

**Recursividad**

La recursividad o recursión es la propiedad de un método por la cual puede llamarse a sí mismo; aunque se puede utilizar como una alternativa a la iteración, una recursión es normalmente menos eficiente en términos de tiempo de computadora que una solución iterativa debido a las operaciones auxiliares que implican las invocaciones suplementarias a los métodos; sin embargo, en muchas circunstancias el uso de la recursión permite a los programadores especificar soluciones naturales y sencillas que en caso contrario serían difíciles de resolver; por esta causa la recursión es una herramienta poderosa e importante en resolución de problemas y en programación. Los algoritmos que se resuelven con la estrategia divide y vence utilizan la recursión para su implementación.

* **Métodos recursivos**

Un método recursivo se invoca a sí mismo de forma directa o indirecta; en recursión directa el código del método f() contiene una sentencia que lo evoca, mientras que en recursión indirecta el método f() invoca a g() el cual invoca a su vez a p() y así sucesivamente hasta que se llama de nuevo al método f().

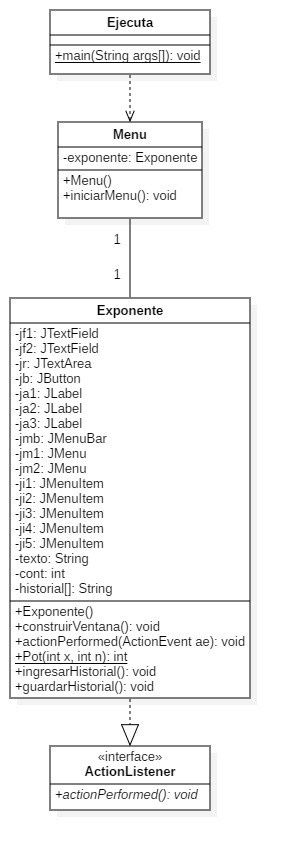
Un requisito para que un algoritmo recursivo sea correcto es que no genere una secuencia infinita de llamadas sobre sí mismo; cualquier algoritmo que origine una secuencia de este tipo no puede terminar nunca, en consecuencia la definición recursiva debe incluir una condición de salida que se llama componente base, en el que f(n) se defina directamente; es decir, no recursivamente, para uno o más valores de n; en definitiva, debe existir posibilidad de salir de la secuencia de llamadas recursivas.

* **Recursión versus iteración**

Tanto la iteración como la recursión se basan en una estructura de control: la iteración utiliza una estructura repetitiva y la recursión, una estructura de selección. Tanto la iteración como la recursión implican repetición: la primera utiliza explícitamente una estructura repetitiva mientras que la segunda consigue la repetición mediante llamadas seguidas al método. La iteración y recursión suponen, cada una, un test de terminación o condición de parada; la iteración termina cuando la condición del bucle no se cumple mientras que la recursión finaliza cuando se reconoce un caso base o la condición de parada se alcanza.

La recursión tiene muchas desventajas; se invoca repetidamente al mecanismo de llamadas a métodos y, en consecuencia, se necesita un tiempo suplementario para realizar cada llamada; esta característica puede resultar cara en tiempo de procesador y espacio de memoria.

**DIAGRAMA UML**



**CÓDIGO DEL PROGRAMA**

**Ejecuta.java**

// Fecha: 07 / Mayo / 2021

// Objetivo: Calcular un exponente.

// Elaboro Equipo 4:

// 1) Flores Navarro Emmanuel

// 2) Ramírez Morales Erick Brian

//Inicio de la clase

public class Ejecuta {

public static void main(String args[]) {

//Variable de instancia

Menu menu = new Menu();

//Acceder a un método

menu.iniciarMenu();

}

}

//Fin de la clase

**Exponente.java**

// Fecha: 07 / Mayo / 2021

// Objetivo: Calcular un exponente.

// Elaboro Equipo 4:

// 1) Flores Navarro Emmanuel

// 2) Ramírez Morales Erick Brian

//Inicio de la clase

import java.awt.event.\*;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

public class Exponente implements ActionListener {

//Atributos

private JTextField jf1;

private JTextField jf2;

private JTextArea jr;

private JButton jb;

private JLabel ja1;

private JLabel ja2;

private JLabel ja3;

private JMenuBar jmb;

private JMenu jm1;

private JMenu jm2;

private JMenuItem ji1;

private JMenuItem ji2;

private JMenuItem ji3;

private JMenuItem ji4;

private JMenuItem ji5;

private String texto;

private int cont = 0;

//Arreglo

private String historial[] = new String[100];

//Método constructor

public Exponente() {}

//Metódo definido por el programador

public void construirVentana() {

//Variables de instancia

JPanel jp = new JPanel();

JFrame jf = new JFrame();

//Barra de herramientas

jmb = new JMenuBar();

jmb.setBounds(0, 0, 110, 20);

jf.add(jmb);

//Herramienta 1

jm1 = new JMenu("Historial");

jmb.add(jm1);

//Opcion 1

ji1 = new JMenuItem("Ver");

jm1.add(ji1);

ji1.addActionListener(this);

//Opcion 2

ji2 = new JMenuItem("Borrar");

jm1.add(ji2);

ji2.addActionListener(this);

jm1.addSeparator();

//Opcion 3

ji3 = new JMenuItem("Guardar");

jm1.add(ji3);

ji3.addActionListener(this);

//Herramienta 2

jm2 = new JMenu("Ayuda");

jmb.add(jm2);

//Opcion 1

ji4 = new JMenuItem("Tutorial");

jm2.add(ji4);

ji4.addActionListener(this);

jm2.addSeparator();

//Opcion 2

ji5 = new JMenuItem("Acerca de...");

jm2.add(ji5);

ji5.addActionListener(this);

//Etiqueta 1

ja1 = new JLabel("Ingrese la base:");

ja1.setBounds(10,30,150,30);

jf.add(ja1);

//Etiqueta 2

ja2 = new JLabel("Ingrese el exponente:");

ja2.setBounds(10,70,150,30);

jf.add(ja2);

//Etiqueta 3

ja3 = new JLabel("Resultado:");

ja3.setBounds(10,110,150,30);

jf.add(ja3);

//Campo de texto 1

jf1 = new JTextField();

jf1.setBounds(160,30,160,30);

jf.add(jf1);

//Campo de texto 2

jf2 = new JTextField();

jf2.setBounds(160,70,160,30);

jf.add(jf2);

//Area de texto

jr = new JTextArea();

jr.setBounds(160,110,160,30);

jf.add(jr);

jr.setEditable(false);

//Boton

jb = new JButton("Resultado");

jb.setBounds(120,160,100,30);

jf.add(jb);

//Presionar

jb.addActionListener(this);

//Ventana

jp.setBounds(20,20,350,250);

jp.setLayout(null);

jf.setLayout(null);

jf.setSize(350,250);

jf.add(jp);

jf.setVisible(true);

jf.setTitle("EXPONENTE");

jf.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

//Metódo definido por el programador

@Override

public void actionPerformed(ActionEvent ae) {

//Atributos

int x, y, r;

//Iniciar excepcion

try {

//Validar boton presionado

if(ae.getSource() == jb) {

//Ingresar valores

x = Integer.parseInt(jf1.getText());

y = Integer.parseInt(jf2.getText());

//Acceder a los métodos

r = Pot(x,y);

jr.setText(r+"");

ingresarHistorial();

}

if(ae.getSource() == ji1) {

//Ciclo

texto = "Historial:";

for(int i=0; i<cont; i++)

texto += "\n"+historial[i];

//Mostrar ventana

JOptionPane.showMessageDialog(null,texto);

}

if(ae.getSource() == ji2) {

//Mostrar ventana

JOptionPane.showMessageDialog(null,"Historial borrado");

cont = 0;

}

if(ae.getSource() == ji3)

guardarHistorial();

if(ae.getSource() == ji4) {

//Mostrar ventana

JOptionPane.showMessageDialog(null,"Pasos a seguir:"

+"\n1) Ingrese un valor en los primeros dos campos de texto."

+"\n2) Presione el boton Resultado."

+"\n\nNota:"

+"\n - Si no hay un valor numerico en los primeros dos campos de texto, el resultado sera un aviso de error.");

}

if(ae.getSource() == ji5) {

//Acceder a un método

JOptionPane.showMessageDialog(null,"Ejercicio 8 - Exponente:"

+"\n - Equipo 4:"

+"\n + Autor 1: Flores Navarro Emmanuel.......201929142"

+"\n + Autor 2: Ramírez Morales Erick Brian...201955520"

+"\n - Fecha: 07 / Mayo / 2021"

+"\n - Derechos Reservados");

}

} catch(NumberFormatException nfe) {

//Capturar e imprimir error

jr.setText("Faltan valores numericos");

} catch(Exception e) {

//Capturar e imprimir error

jr.setText("Error");

}

}

//Metódo definido por el programador

public static int Pot(int x, int n) {

//Validar valor

if(n==0)

//Retorno

return 1;

else

//Retorno

return(x\*Pot(x,n-1));

}

//Metódo definido por el programador

public void ingresarHistorial() {

historial[cont] = jf1.getText()+"^"+jf2.getText()+" = "+jr.getText();

cont++;

}

//Metódo definido por el programador

public void guardarHistorial() {

//Variables de instancia

FileWriter fw = null;

PrintWriter pw = null;

//Atributo

texto = "";

//Iniciar excepcion

try {

//Variables de instancia

fw = new FileWriter("Exponente.txt");

pw = new PrintWriter(fw);

//Ciclo

for(int i=0; i<cont; i++)

texto += "\n "+historial[i];

//Guardar mensajes y resultado

pw.println("\n ========== EXPONENTE ==========");

pw.println(texto);

pw.println("\n ===============================\n");

} catch(Exception e1) {

//Capturar e imprimir error

JOptionPane.showMessageDialog(null,"ERROR..."+e1.getMessage()+"...","ERROR",JOptionPane.WARNING\_MESSAGE);

} finally {

//Iniciar excepcion

try {

//Validar dato

if (null != fw)

fw.close();

//Mostrar ventana

JOptionPane.showMessageDialog(null,"Historial guardado");

} catch(Exception e2) {

//Capturar e imprimir error

JOptionPane.showMessageDialog(null,"ERROR..."+e2.getMessage()+"...","ERROR",JOptionPane.WARNING\_MESSAGE);

}

}

}

}

//Fin de la clase

**Menu.java**

// Fecha: 07 / Mayo / 2021

// Objetivo: Calcular un exponente.

// Elaboro Equipo 4:

// 1) Flores Navarro Emmanuel

// 2) Ramírez Morales Erick Brian

//Inicio de la clase

import javax.swing.JOptionPane;

public class Menu {

//Variable de instancia

private Exponente exponente = new Exponente();

//Método constructor

public Menu() {}

//Metódo definido por el programador

public void iniciarMenu() {

//Atributos

String panel;

int opc;

//Inicio Menu

do{

panel = JOptionPane.showInputDialog(null,"Ingrese el numero de la opcion a ejecutar:"

+"\n 1) Inicio"

+"\n 2) Salir","MENU",JOptionPane.QUESTION\_MESSAGE);

//Iniciar excepcion

try {

//Ingresar un valor

opc = Integer.parseInt(panel);

} catch(Exception e) {

//Capturar y mostrar error

JOptionPane.showMessageDialog(null,"ERROR..."+e.getMessage(),"ERROR",JOptionPane.WARNING\_MESSAGE);

opc = 0;

}

switch(opc) {

case 2 -> {

//Mostrar mensaje y cerrar panel

JOptionPane.showMessageDialog(null,"Gracias por usar este programa, hasta luego!!!");

System.exit(0);

}

case 1 -> {

//Acceder a un método

exponente.construirVentana();

opc = 2;

}

default -> {

//Mostrar mensaje

JOptionPane.showMessageDialog(null,"Opcion no valida","ERROR",JOptionPane.WARNING\_MESSAGE);

}

}

}while(opc != 2);

//Fin Menu

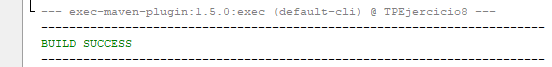
}

}

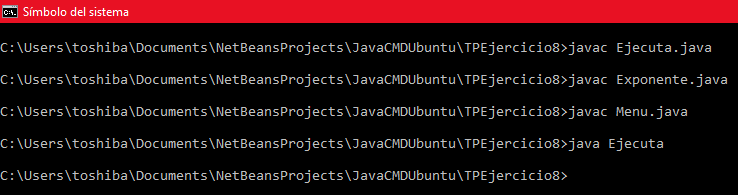
//Fin de la clase

**CAPTURAS DE PANTALLA**

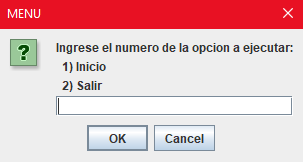
**NetBeans**

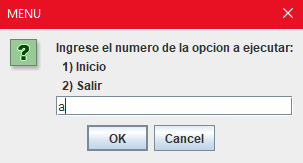


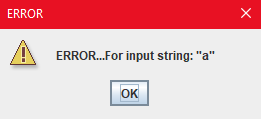
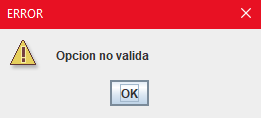
**CMD**

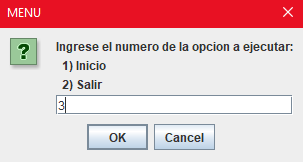


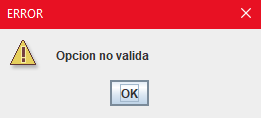
**Ventanas de ejecución**

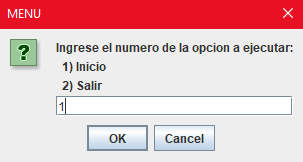


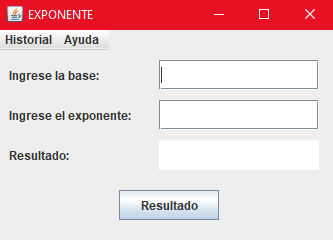


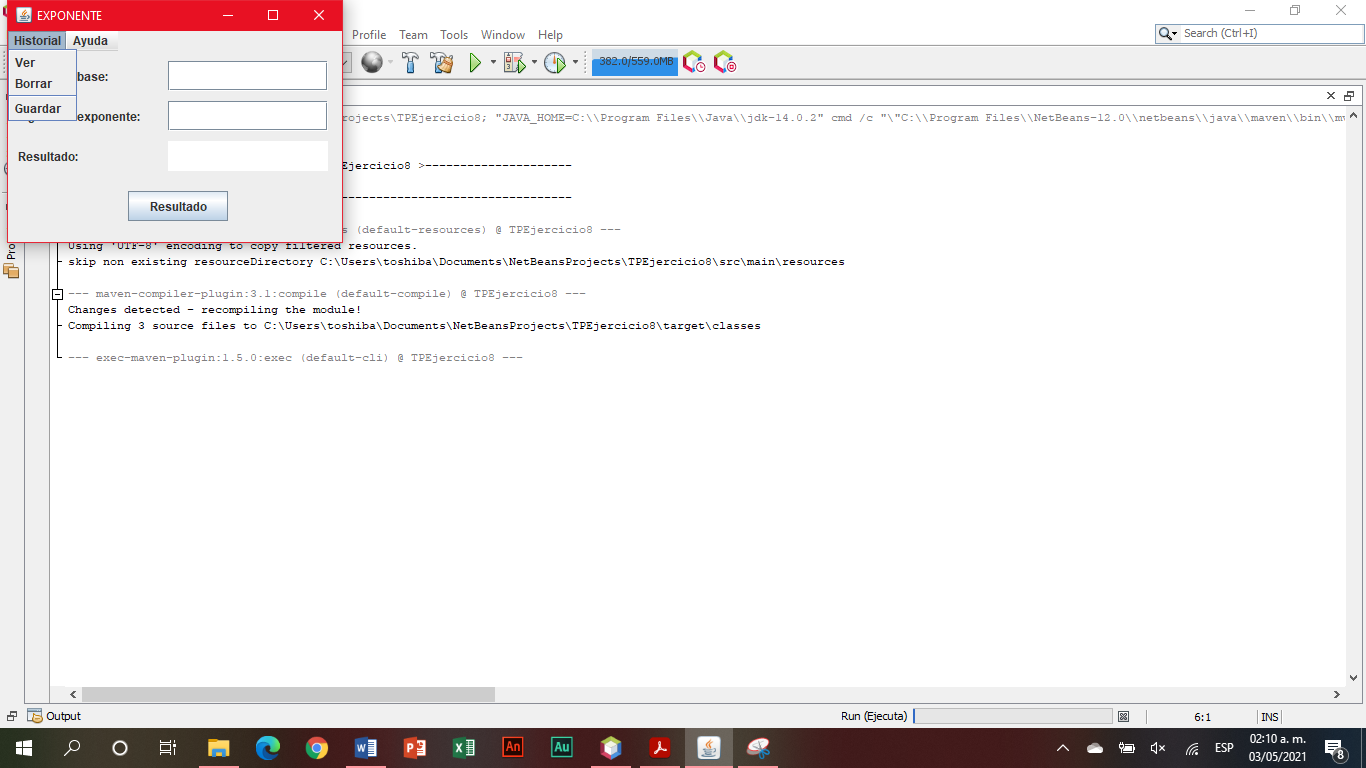
 

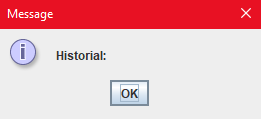


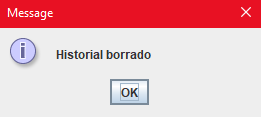


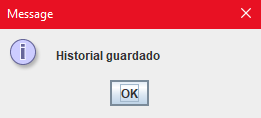


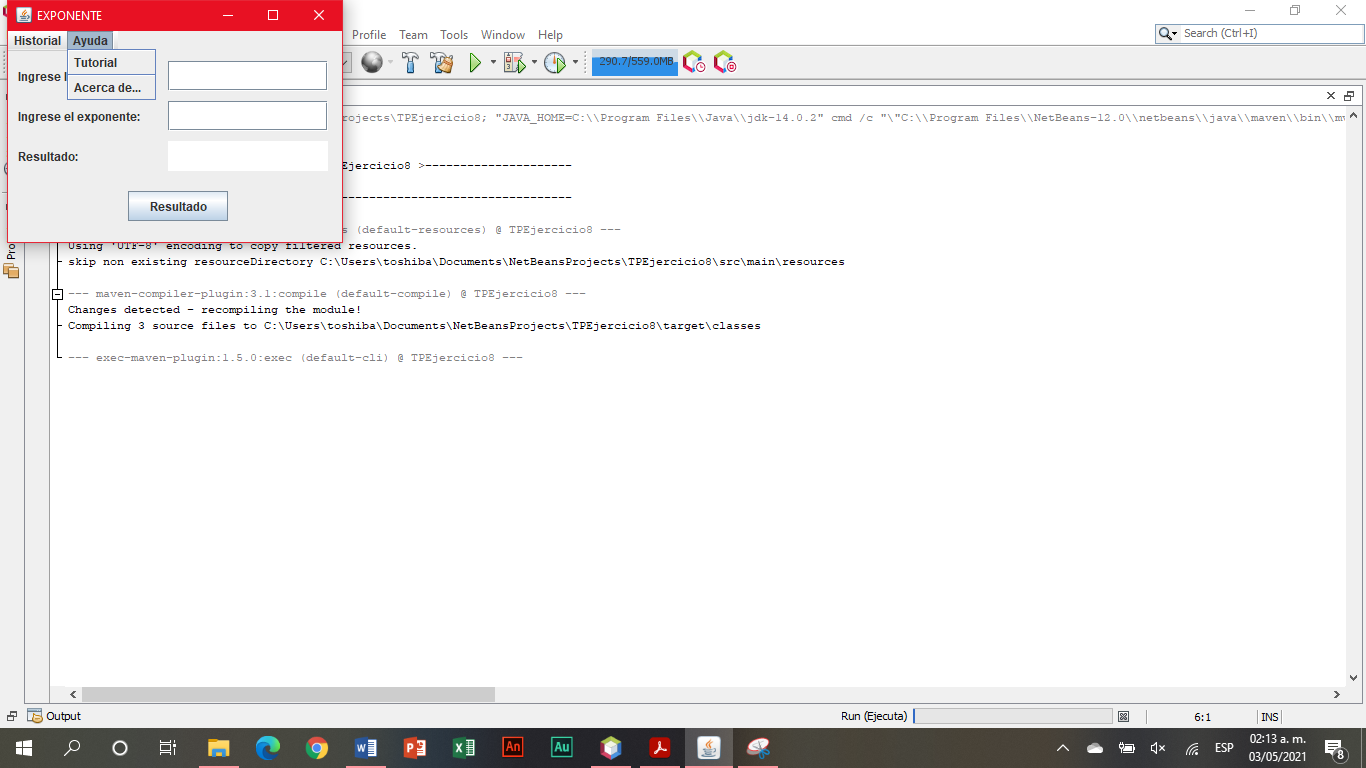


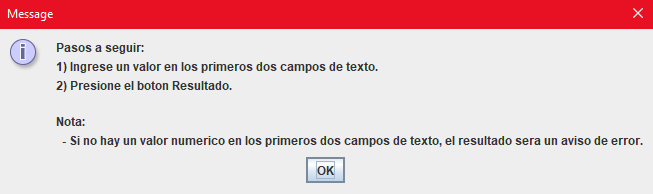


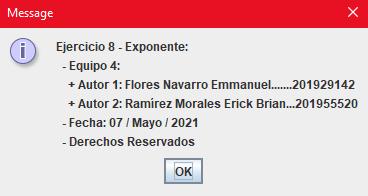


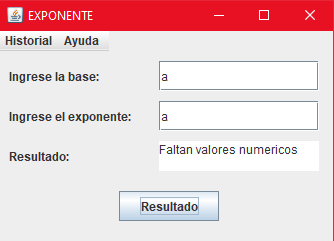


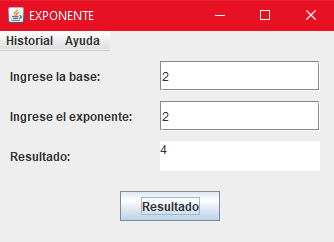


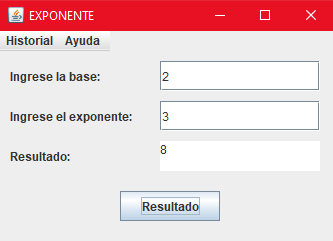


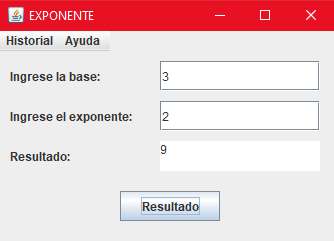


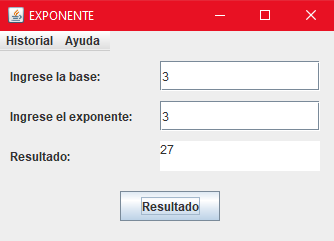


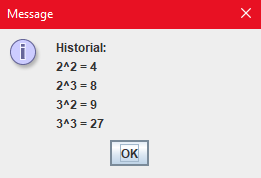


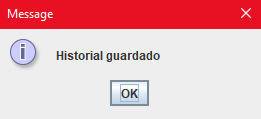


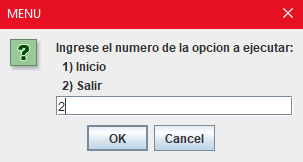


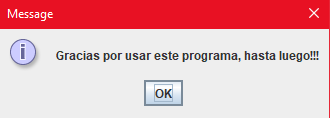




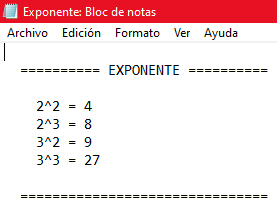








**Exponente.txt**



**CONCLUSIÓN**

El uso de la recursión permite a los programadores especificar soluciones naturales y sencillas que en caso contrario serían difíciles de resolver; por esta causa la recursión es una herramienta poderosa e importante en resolución de problemas y en programación.

# **BIBLIOGRAFÍA**

Céron, C. (27 de Abril de 2021). *Unidad5 \_ Recursividad*. Obtenido de Blackboard.

Joyanes, L., & Zahonero, I. (2011). Programación en Java 6. En L. Joyanes, & I. Zahonero, *Programación en Java 6* (págs. 453-459). México, D.F: Mc Graw Hill.

**PROGRAMACIÓN II**

**RÚBRICA HOJA DE EVALUACIÓN DE PRÁCTICAS**

MATRICULA: 201955520 FECHA: 07 / Mayo / 2021

NOMBRE: Ramírez Morales Erick Brian No. PRACTICA: Ejercicio 8 INDIVIDUAL ( ) COLABORATIVA ( X )

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Deficiente** | **Suficiente** | **Bueno** | **Excelente** | **Calificación Obtenida** | |
| **CRITERIOS** | **1-5.9** | **6‐7.9** | **8‐9** | **9.1‐10** |
| **%** | **Puntos** |
| **CONOCIMIENTO TEORICO**  **20%** | Conocimiento deficiente de los fundamentos teóricos de POO y no puede aplicarlos en el diseño de clases. | Conocimiento confuso de los fundamentos teóricos de POO y el diseño de las clases y relaciones es incompleto. | Conocimiento claro de los fundamentos teóricos POO pero requiere mejorar el modelado de las clases, sus métodos y sus relaciones entre clases. | Dominio del Conocimiento de los fundamentos teóricos POO y puede aplicarlos de forma completa en el modelado de las clases, métodos y todas las relaciones entre clase. |  |  |
| **EJECUCCIÓN DE LA PRACTICA**  **30%** | No puede realizar la práctica ya que desconoce el entorno de trabajo y desarrollo de la práctica en lenguaje UML y Java. | Realiza la práctica de forma incompleta ya que desconoce el entorno de trabajo del lenguaje UML y Java | Realiza la práctica, pero requiere mejorar en el manejo del entorno de trabajo del lenguaje de programación (sintaxis y  semántica) | Realiza la práctica de forma correcta y completa, demuestra dominio del entorno de trabajo del lenguaje de programación (sintaxis y semántica). |  |  |
| **SOLUCIÓN DE LA PRACTICA**  **40%** | No puede generar las soluciones o programas a los problemas planteados ya que no posee el dominio teórico y práctico del modelado y el lenguaje de programación. | Propone soluciones confusas o programas incompletos a los problemas planteados, ya que carece del dominio del modelado y lenguaje de programación | Genera soluciones con poca profundidad y los programas no están orientados de acuerdo a los problemas solicitados, por lo cual no tiene un dominio profundo de la temática y del Lenguaje. | Genera soluciones con profundidad y los programas son correctos de acuerdo a los problemas planteados, por lo cual demuestra un dominio de la temática y del Lenguaje de Programación. |  |  |
| **ACTITUD DE APRENDER Y COLABORAR EN EQUIPO DE TRABAJO**  **10%** | No posee una actitud proactiva para un aprendizaje autónomo y no le gusta participar y trabajar en equipo. | Posee una actitud propositiva para un aprendizaje autónomo, participa pero no le gusta trabajar en equipo. | Posee una actitud propositiva logrando un aprendizaje autónomo, colaborativo, le gusta trabajar en equipo pero requiere mejorar su participación y portaciones de forma profunda. | Posee una actitud proactiva logrando un aprendizaje autónomo, participa con propuestas concretas y profundas, le gusta trabajar en equipo y asume su responsabilidad dentro para lograr éxito del equipo |  |  |
| **Total**  Equipo 4 | | | | |  |  |

****

**Firma del Alumn@**

**Vo. Bo. Docente**